Wat is er allemaal aan de hand? Introductie van Real Time Intelligence en Complex Event Processing

In de wereld om ons heen gebeurt van alles. Voortdurend. In de fysieke wereld en misschien nog wel meer in de computersystemen die de wereld waarnemen en voor een deel besturen. Grote gebeurtenissen met hoge zichtbaarheid en soms enorme impact. En heel, heel veel kleine gebeurtenisjes. Vaak onbeduidend. Maar soms helemaal niet onbeduidend – soms is één van miljoenen waarnemingen of registraties van heel groot belang. Een indicatie van file of brand, van een instortende beurskoers of een credit card fraudegeval. Vaak trouwens kan zo'n conclusie pas worden getrokken als er meerdere van die kleine bevindingen met elkaar in verband worden gebracht en een patroon of trend kan worden afgeleid.

Dit artikel introduceert het concept 'Complex (soms ook Intelligent) Event Processing' (CEP of IEP) – het terrein van het analyseren van grote aantallen, voortdurend binnenstromende meldingen van op zichzelf mogelijk betekenisloze gebeurtenissen. Met CEP hebben we een methode om naalden in event-hooibergen te vinden –bevindingen met een concrete betekenis, waarop analyses, conclusies, beslissingen en acties kunnen worden gebaseerd.

CEP wordt vaak geïntegreerd in Java infrastructuren en applicaties. Favoriete bronnen van events en bestemmingen van CEP uitkomsten zijn ondermeer JMS, JMX, HTTP en (MD) EJB. Veel van de verwerking wordt gedaan met behulp van XML en Java en Java frameworks – bijvoorbeeld JAXB, JSON maar ook het Spring Framework – en interactie met relationele databases is veelal via JDBC en eventuele persistency frameworks. CEP engines kunnen binnen JEE applicatie servers worden gerund, maar veelal ook op zichzelf staand.

De zoekvragen voor CEP worden in een speciale taal beschreven – de EPL of Event Processing Language. Er is echter niet een standaard EPL, er is een flink aantal EPLs van verschillende organisaties en leveranciers. Sommige van deze EPLs zijn afgeleid van of geïntegreerd met Business Rule Languages – zoals Tibco BEPL en JBoss DRL. Andere EPLs zijn geïnspireerd door, afgeleid van of zelfs geïntegreerd met SQL, zoals CCL, StreamSQL, EQL en CQL. Het blijkt handig als in één query zowel event-streams als relationele data-bronnen kunnen worden gecombineerd, bijvoorbeeld om historische en referentie data te combineren met de live-event-feed . EPL queries sturen de event processing engines aan die we kunnen inzetten om events te verwerken en te promoveren tot bruikbare, betekenisvolle meldingen.

In dit artikel kijken we naar de concepten en bedoelingen achter CEP, de beschikbare implementaties van de CEP engines en enkele voorbeelden van het toepassen van CEP en CQL.

Uitgangspunten en Definities

Hierboven gaf ik aan dat er een speciale query-taal is voor events, verwant aan SQL. Maar waarom zijn SQL en bestaande relationele (eventueel in memory) databases niet goed genoeg voor het analyseren van de event-data?

Een goede vraag – zelfs al is hij retorisch. En een die raakt aan het wezen van CEP. Events zijn over het algemeen klein – met een hele kleine payload. Ze komen normaalgesproken met niet meer dan een tijdstip, een vorm van identificatie van de bron van het event (te gebruiken voor correlatie) en een meet-waarde. Vaak – als we speciale patronen of afwijkingen proberen te ontdekken – zijn de meeste events zelfs compleet irrelevant. Alleen: events komen in enorme aantallen, tot duizenden of zelfs honderdduizenden per seconde. En ze blijven komen. Het analyseren van events is niet een kwestie van eenmalig een zoekvraag uitvoeren tegen een statische set data – zoals met SQL tegen een relationele database. Omdat zodra de vraag gesteld en beantwoord is er alweer een nieuwe lading events is ontvangen zou dezelfde vraag steeds een ander antwoord kunnen opleveren. De vraag moet voortdurend gesteld blijven worden omdat de events voortdurend blijven komen. En dat is het bijzondere aan C (*continuous*) QL. De vraag wordt niet gekoppeld aan een statische tabel met data en eenmalig beantwoord, maar aan een 'stream' met voortdurend binnenkomende events.

Over het algemeen wordt in een CQL query een 'time window' gedefinieerd, een periode van enkele milliseconden tot normaalgesproken niet meer dan minuten of uren maar soms tot wel maanden waarover geaggregeerd of geanalyseerd wordt. CQL queries worden gestart en blijven uitgevoerd worden tot ze expliciet weer worden gesloten – niet tot het eerste antwoord is opgeleverd.

Het real-time karakter van niet alleen het arriveren van de events maar ook van het detecteren van afwijkingen en overschrijdingen van grenswaarden, patronen, aggregaties en specifiek gefilterde gebeurtenissen is kenmerkend voor CEP. De term Real Time Intelligence wordt soms gehanteerd om aan te geven dat de real time verwerking van event stromen moet leiden tot directe inzichten. De CEP engine verwerkt de continue stromen en produceert – in een uiteraard veel lager volume – bevindingen in de vorm van 'business events'.

Deze laatste events zijn vaak aanzienlijk omvangrijker dan de signalen die de CEP engine ingaan. Ze beschrijven een antwoord op een CQL query met veelal een vorm van samenvatting van de verwerkte (relevante) event-data en de bevindingen. CEP engines sturen hun uitkomsten vaak naar een BAM (Business Activity Monitoring) tool dat ze bijvoorbeeld publiceert op een dashboard of vertaalt in alerts die verzonden worden. Ook is een Enterprise Service Bus een typisch doel voor een CEP engine, bijvoorbeeld als ingangspunt voor een EDA (Event Driven Architecture) met SOA services die door bepaalde business events worden getriggerd.

Overigens: de events die worden verwerkt door de CEP zijn over het algemeen van een heel andere orde – veel kleiner en betekenislozer en veel frequenter – dan het type event dat in een Event Driven Architecture de 'interactie' tussen processen en services verzorgt.

Toepassingen van en doelen met CEP

De belangrijkste en vroegste toepassing van CEP wordt gevonden in grote financiële instellingen waar complexe event verwerking de automatische aankoop en verkoop van aandelen, valuta en futures (op grondstoffen) aanstuurt. Zodra bijvoorbeeld een prijs trend wordt waargenomen in een van de vele koersen door analyse van de continue stroom van koerswijzigingen kan daarop met aan- of verkoop op worden gereageerd. Ook in andere termijnmarkten – zoals sinds kort nadrukkelijk die

voor energie – worden deze mechanismen en algoritmes toegepast. CEP wordt hier vooral ingezet voor patroonherkenning: een koersontwikkeling in een bepaalde richting.

Een ander belangrijk fenomeen bij CEP is de detectie van 'non-events': events die *niet* optreden – maar die wel hadden moeten optreden. Dit speelt bijvoorbeeld bij reizigers die een land binnenkomen op een visum dat drie maanden geldig is en waarvoor het exit-event niet gevonden wordt, maar ook bij een koffer die bij de check in balie langs de RFID sensor gaat maar niet binnen een vastgestelde periode bij de bagage afhandelaars arriveert. Of een security badge die de beveiligde zone binnengaat maar er niet binnen een bepaalde periode weer uitkomt.

Het bewaken van Service Level Agreements – zowel in de dienstverlening bij call-centers als de beschikbaarheid van mobiele telefoondiensten als de up-time en response-tijden van (gehoste) applicaties en web services wordt veelvuldig gebruik gemaakt van CEP. DNS attacks – te frequente requests van bepaalde IP adressen – kunnen met CEP vroegtijdig gedetecteerd worden. Net als onregelmatigheden in hartslag of temperatuur bij de monitoring van patiënten.

In het verkeer zijn vele voorbeelden van CEP te vinden: traject(snelheids)controle, spookrijder-detectie, automatische afhandeling van betaald parkeren en rekeningrijden. Ook het gebruik van de OV Chipkaart zou middels CEP kunnen worden opgelost.

Er is een aanzienlijke subcategorie van toepassingen waar naast patroonherkenning op basis van tijd ook de locatie een belangrijke rol speelt. Credit card fraude detectie bijvoorbeeld zoekt ondermeer naar betalingen met dezelfde kaart op verschillende plaatsen binnen een te kort tijdbestek. Ook wordt CEP gebruikt om SMS berichten te sturen naar mobiele telefoons die een ander land zijn binnengegaan. Ook andere gelocaliseerde services – welke aanbiedingen vind je in een straal van 10 meter om de plek waar je je nu bevindt – worden op basis van CEP aangeboden.

In al deze toepassingen komen verschillende aspecten steeds terug:

1. events worden bekeken, gefilterd op eigenschappen, en vergeleken (gecorreleerd) over een bepaalde periode (time window)
2. over events die bij elkaar horen (inclusief de ontbrekende non-events) worden berekeningen, aggregaties en patroonherkenning uitgevoerd
3. als de uitkomsten voldoen aan vooraf bepaalde regels – overschrijding van grenswaarden, verwacht event ontbreekt na de maximale wachtperiode, specifiek patroon treedt op – wordt een resultaat event gegenereerd en naar een afnemer verzonden

Opbouw van CEP applicaties

CEP applicaties moeten onze systemen voorzien van relevante signalen waarop actie moet worden opgenomen. De actie kan bijvoorbeeld bestaan uit een refresh van een User Interface maar ook de aanroep van services, het versturen van berichten of alerts in verschillende vormen. De CEP applicatie is hiervoor vaak via een Adapter of Connector aangesloten op uitgaande communicatie-mechanismes als JMS, JMX en HTTP maar ook doelen als JDBC en Files. Vergelijkbare Adapters ofConnectors worden aan de inkomende kant van de CEP applicatie gebruikt om event-bronnen af

te tappen. Sommige CEP producten bieden specifieke Connectors/Adapters voor specifieke event-feeds van financiële bronnen als Thomson Reuters, Bloomberg en FX.

In het hart van de CEP applicatie zit een Processor die de daadwerkelijke event-verwerking uitvoert – correlatie, schuivend time-window, filtering, aggregatie, patroonherkenning – en eventueel de uitgaande events produceert met de benodigde payload. De programmering van de Processor wordt gedaan met EPL instructies, zoals EQL, CCL en CQL queries.

Een voorbeeld van een CQL query die een event produceert als er een wijziging optreedt in het aantal events in de laatste vijf seconden waarvoor geldt dat het eenOfAndereWaarde property uit de payload van het event groter is dan 15:

```
select count(eenOfAndereWaarde) as countEvents
from   eventChannelToProcessor [range 5]
where  eenOfAndereWaarde > 15
```

De volgende CQL query produceert een event als er een voorbeeld wordt gevonden van een event dat een lagere waarde heeft voor eenOfAndereWaarde dan het event dat er net aan vooraf ging:

```
select values.thevalue as deTrendBreukWaarde
from   eventChannelToProcessor
MATCH_RECOGNIZE(
  MEASURES A.eenOfAndereWaarde as thevalue
  PATTERN (A)
  DEFINE
   A as (A.eenOfAndereWaarde < prev(A.eenOfAndereWaarde ))
) as values
```

Tussen adapters en processors bevinden zich channels - de transportpijpen waarlangs de events vervoerd worden. Deze pijpen kunnen ondermeer caching, heartbeat, throttling (voorkomen van overstromen en vastlopen van de processor door te grote aantallen events), filtering en timestamping diensten uitvoeren.

Een processor kan events lezen uit meerdere channels, bijvoorbeeld een mix van streams (continue events) en relations (database bronnen). Meerdere listener-channels kunnen op een processor worden aangesloten – zodat de geproduceerde events op verschillende manieren hun weg vinden.

In meer complexe situaties zullen trouwens CEP applicaties uitgroeien to grotere Event Processing Networks waarin meerdere processors worden ingezet die deels op elkaar aansluiten en deels extra kanalen introduceren. De hier getoonde afbeelding wordt dan uitgebreid met extra elementen die inhaken op de afgebeelde processor en channels.

CEP Implementaties

Er is meer dan een tiental CEP implementaties beschikbaar waarmee de verwerking van events en eventueel de aansluiting op BAM, EDA en SOA kan worden verwezenlijkt. Al grote spelers in de

industrie bewegen zich inmiddels op dit vlak – sommige al heel lang, andere pas recent – en er lijkt een verdere consolidatie voor de hand te liggen.

In Augustus 2009 publiceerde het marktanalyse bureau Forrester een rapport over CEP producten. De belangrijkste conclusies – zoals altijd tamelijk meervoudig uitlegbaar: Progress Software (Apama) en Aleri zijn de leiders. IBM (WebSphere Business Events, System S/InfoSphere Streams en tot op zekere hoogte de ILog rule engine), StreamBase Systems, en TIBCO Software (Tibco Business Events) zijn ook onderdeel van de Leaders' circle. Coral8 (recent verworven door Aleri), Oracle (met name gebaseerd op BEA's CEP producten) en UC4 (Senactive) zijn in Forrester termen *Strong Performers* die op korte termijn ook Leaders kunnen worden. Datzelfde geldt voor EsperTech – de enige open source oplossing die is meegewogen. Andere open source producten zoals OpenESB IEP (Glassfish) en JBoss Drools Fusion zijn niet beoordeeld. Ook Microsoft met haar recente StreamInsight initiatief wordt door Forrester niet eens genoemd.

De eenvoudigste manier om met CEP aan de slag te gaan is waarschijnlijk door een van de open source producten te proberen – OpenESB IEP en EsperTech zijn dan meest voordehandliggende opties. Hoewel de concepten van de meeste CEP platforms erg op elkaar lijken, zijn applicaties zeker niet onderling uitwisselbaar. Het ontbreken van een gestandaardiseerde EPL staat dat vooralsnog in de weg. Een zekere convergentie van EPLs die van SQL zijn afgeleid lijkt zich wel aan te dienen, maar vooralsnog met kleine stapjes. Bekende voorbeelden van met SQL verwante EPLs zijn ondermeer CQL en RAPIDE (Stanford University), CCL (Aleri Corel8), StreamSQL (StreamBase) en EQL (Esper)

Het meest vergaande voorbeeld van de integratie tussen SQL en EPL is CQL – Continuous Query Language. CQL kent zijn oorsprong op de Universiteit van Stanford, met Cambridge en CalTech een van de pioniers op het terrein van CEP. CQL is een uitbreiding op SQL, waarin de zoekvragen met betrekking tot de events en uitgevoerd tegen streams, kunnen worden gedefinieerd. Een CQL query bevat naast de speciale event gerelateerde syntax en operatoren ook alle normale SQL operaties. CQL is geselecteerd door ondermeer SUN Microsystems, BEA en Oracle voor hun respectievelijke CEP engines (Glassfish OpenESB IEP en Oracle CEP).

Conclusie

De behoefte aan real-time verwerking van grote aantallen gebeurtenissen en feitjes tot conclusies en triggers (tot actie) heeft geleid tot de ontwikkeling van Complex Event Processing (CEP) als een onderzoeksterrein en marktsegment. CEP verbindt de real time wereld van transacties, data feeds, RFID en sensor waarden en web click events op een intelligente manier met ondermeer BAM, SOA en BPM/Workflows. CEP engines verwerken inkomende event streams en gebruiken correlatie, filtering, aggregatie en patroonherkenning om tot afgeleide en betekenisvolle business events te komen. CEP engines kunnen uitstekend worden geïntegreerd met Java applicaties en infrastructuren; veel gebruikte Java mechanismen zijn JMS,JMX, JDBC en POJOs.

In deel twee van dit artikel zal een aantal voorbeelden van CEP toepassing in meer detail worden uitgewerkt met verschillende CEP engines.

Voor meer informatie over CEP en implementaties, Google op CEP of kijk op http://complexevents.com/.